ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

Visual Intelligence in Resource-Constrained Edge Devices: A Review of Raspberry Pi

Archana D.[1]; Radhika V.[2]

[1,2]Department of Biomedical Engineering Sri Ramakrishna Engineering College Coimbatore, India

Publication Date: 2026/03/11

Abstract: Developments in computer vision have provided a great deal of capability to smart systems to sense the world around them and be able to make autonomous decisions with a broad spectrum of uses. Nonetheless, executing the vision-based algorithms in resource-constrained edge devices is not easy since their capabilities are severely limited in terms of computational and memory resources, as well as energy supply. This review explores the computer vision algorithms that can be applied to low-cost and low-power embedded systems with a specific focus on the Raspberry Pi family. Methods of object detection and multi-object tracking (including classical vision algorithms, as well as deep learning models like YOLO, SSD, and MobileNet) are reviewed and compared with each other in terms of the indicators of the evaluation characteristics that are commonly reported (such as inference speed, latency, model size, number of parameters, and detection accuracy (mAP)) when available. Also, the techniques of optimisation like quantization, pruning, lightweight backbone design, and embedding edge AI accelerators (e.g., Google Coral TPU and Intel Neural Compute Stick 2) are discussed. This survey summarizes the results of benchmark data sets and real-world systems to highlight major trade-offs in embedded vision implementation and suggests real- world combinations of detectors and trackers to use in appli- cations of surveillance, mobile robots, and smart transportation

Keywords: *Edge Computing, Raspberry Pi, Object Detection, Object Tracking, Resource-Constrained Devices, Lightweight Vision Algorithms.*

How to Cite: Archana D.; Radhika V. (2026) Visual Intelligence in Resource-Constrained Edge Devices: A Review of Raspberry Pi. *International Journal of Innovative Science and Research Technology*, 11(3), 187-199. https://doi.org/10.38124/ijisrt/26mar114

I. INTRODUCTION

Computer vision has become a fundamental enabling tech- nology in recent intelligent systems, and is used in applications like autonomous robotics, unmanned aerial vehicles (UAVs), intelligent surveillance, assisted healthcare monitoring, intel- ligent transportation, and industrial automation. Traditional computer vision pipelines have been mostly designed to run on an expensive high-performance workstation or cloud server, and in this case, there is a mega-load of computational re- sources and memory. Conversely, recent developments in edge computing continue to focus more on performing perception functions on embedded systems. The low-latency response, reduced reliance on network connectivity, enhanced privacy preservation, as well as constant operation in dynamic real- world settings are made possible by this paradigm.

The Raspberry Pi family has become a prominent topic among low-cost single board computers (SBCs) as a practical edge platform because of its small size, low cost, and wide ecosystem. Raspberry Pi devices are capable of serving a large variety of vision workflows, including basic image processing; when models are optimized correctly, deep learning inference is possible. Still, implementing vision applications at the edge is not so straightforward, as Raspberry Pi boards have very limited capabilities in terms of CPU performance, memory bandwidth, storage, thermal (throttling), and power consump- tion. These constraints are especially important in tasks that are computationally challenging, like real-time object detec- tion and multi-object tracking, which tend to demand steady frame-rate processing.

Histogram-based analysis, edge detection, background sub- traction, contour-based segmentation, and correlation filter tracking algorithms such as KCF, MOSSE, and CSRT are also classical methods of computer vision that are still useful on resource constrained platforms. These methods also generally have minimal computation requirements and run-time perfor- mance that is predictable and thus can be used in structured applications like line-following robots, controlled industrial in- spection, and simple tracking problems. Nonetheless, classical methods severely degrade in complicated real-world scenarios that entail occlusions, appearance changes, scale changes, alterations in illumination, and background clutter.

However, perception models using deep learning have proven to be more robust and accurate under various operating conditions. Object detection models like YOLO

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

variants, SSD, MobileNet-SSD, and EfficientDet-Lite that are lightweight have been increasingly studied and evaluated on Raspberry Pi hardware. They must, however, be carefully balanced in terms of accuracy (mAP), throughput (FPS), latency, as well as resource usage (RAM and energy), because of their embedded nature. To help overcome these issues, common optimization strategies such as INT8 quantization, pruning, model compression, reduced input resolution, and efficient inference systems such as TensorFlow Lite, OpenVINO, and ARM NN are often used by researchers. Additionally, external accelerators like Google Coral EdgeTPU and Intel Neural Compute Stick 2 (NCS2) have been identified to provide a significant improvement in inference performance and low power consumption.

Other than detection, object tracking is also required in practical edge vision systems since it provides consistency in identity across frames, enhances temporal accuracy, and minimizes redundant detections. DeepSORT, ByteTrack, and OC-SORT are state-of-the-art multi-object tracking (MOT) algorithms that have been adopted because of their data as- sociation capability. However, tracking adds more complexity to the computation because of motion modeling, bounding- box association, and in certain pipelines, appearance feature extraction. Therefore, a proper choice of detector-tracker pair is critical to the success of real-time operation of Raspberry Pi-based applications.

This review summarizes recent studies on computer vision implementation on Raspberry Pi edge devices, including clas- sical and deep learning-based methods to detect and track objects. The surveyed studies are discussed with the help of conventional performance metrics such as FPS, latency, memory footprint, model size, number of parameters, accuracy (mAP), and measures of tracking quality such as MOTA and IDF1. This aims at summarizing the state of the art, identifying key deployment trade-offs, and offering recommendations on the most efficient pipelines to use in embedded systems in surveillance, assistive robotics, and smart transportation.

II. EDGE HARDWARE PLATFORM: RASPBERRY PI

- Raspberry Pi 3: This is the basic version. It is good for simple programs and small vision tasks, but it is slow for deep learning because RAM and CPU are limited.
- Raspberry Pi 4: This is better than Pi 3. It has more RAM options and faster CPU, so it is suitable for running lightweight object detection and tracking models.
- Raspberry Pi 5: This is the latest and fastest among these three. It gives higher processing speed and better performance for real-time computer vision work, especially when models are heavy.

Table 1 Comparison of Raspberry Pi Hardware Platforms

Feature	Raspberry Pi 3	Raspberry Pi 4	Raspberry Pi 5
CPU	Quad-core ARM Cortex- A53 @ 1.2–1.4 GHz	Quad-core AR Cortex- A72 @ 1.5 GHz	Quad-core ARM Cortex- A76 @ 2.4 GHz
RAM	1 GB LPDDR2	2 GB / 4 GB / 8 GB PDDR4	4 GB / 8 GB LPDDR4X
GPU	Broadcom VideoCore IV	Broadcom VideoCore VI	Broadcom VideoCore VII
AI / Compute	Basic, not AI- accelerated	Moderate inference tasks	Enhanced AI suitability

III. THEORETICAL BACKGROUND: COMPUTER VISION TECHNIQUES

A. Classical Computer Vision Methods

Classical computer vision describes the image and video processing methods which are based on human-defined fea- tures and rule-based decision making, not on learning from data. They are based on mathematical modeling, geometry, and signal processing concepts where domain knowledge is explicitly implemented into the pipeline.

A general classical vision system has a sequential flow comprising image capture, preprocessing, feature extraction, and decision-making. Through common functions, filtering, thresholding, edge detection, corner detection, morphological processing, contour extraction, and optical flow estimation are widely used. Sobel, Canny, Harris corner detector, Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Trans- form (SIFT), Speeded-Up Robust Features (SURF), and Ori- ented FAST and Rotated BRIEF (ORB) are some algorithms that have been extensively used in object detection, tracking, and scene understanding, in which computational resources are often constrained.

- *Classical Object Detection Methods:*

Table 2 is a summary of common classical methods of object detection and their properties.

Table 2 Classical Object Detection Methods

Algorithm	Description
HOG (Histogram of Oriented Gradients)	Computes local histograms of gradient orientations to represent object shape; widely used for pedestrian and rigid object detection.
Haar Cascade Classifiers	Detect objects using Haar-like intensity features and AdaBoost-based cascading; efficient in grayscale and commonly used for face and eye detection.
SIFT / SURF / ORB	Detect and describe scale- and rotation-invariant keypoints for object recognition, matching, and tracking; ORB offers improved computational efficiency.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

Background Subtraction	Models the static background to segment moving foreground objects; effective for video surveillance in fixed-camera environments.
Frame Differencing	Detects motion by subtracting consecutive frames; computationally simple but sensitive to noise and illumination variations.
Edge Detection (Canny, Sobel)	Extracts object boundaries by identifying intensity discontinuities; often used as a preprocessing step for segmentation.
Template Matching	Locates objects by correlating input images with predefined templates; performs best when object pose and scale variations are limited.
Contour Analysis	Identifies object regions using geometric properties such as shape, boundary, and area; performance depends on reliable segmentation quality.

➢ *Classical Object Tracking Methods:*

The goal of classical tracking is to estimate object motion between frames with a geometric or statistical model and it is not always computationally expensive. Table 2 is a summary of popular approaches.

Table 3 Classical Object Tracking Methods

Algorithm	**Description**
Centroid Tracking	Tracks object identity based on centroid displacement; suitable for multiple objects with clear separation.
Kernelized Correlation Filter	Correlation-filter tracker implemented efficiently in the frequency domain; suitable for real-time tracking.
Channel and Spatial Reliability Tracker	Robust correlation-filter tracker handling scale variation and partial occlusions; higher computational cost than KCF.
Kalman Filter	Predicts object state using linear motion models; smooths noisy detection outputs; widely used in MOT pipelines.
Particle Filter	Uses multiple weighted hypotheses for tracking; handles non-linear and non-Gaussian dynamics; computationally heavier.
Optical Flow	Estimates pixel-wise motion vectors between frames; effective for dense tracking but sensitive to lighting variation.
SIFT / SURF / ORB	Tracks by matching keypoints across frames; suitable when strong texture features exist.
Contour Analysis	Tracks objects by changes in contour properties (shape, position, centroid); depends on stable segmentation.

B. Deep Learning-Based Vision Methods

Deep learning has revolutionized computer vision by allow- ing object detection, classification, segmentation, and tracking with high accuracy using both convolutional and transformer- based architectures. Nonetheless, there are tremendous limi- tations to the deployment of deep models on embedded edge platforms, primarily due to limitations in processing power, memory, power consumption, and thermal stability.

In order to assist with edge deployment, studies have high- lighted model adaptation techniques which include: low weight network, low input resolution, quantization (FP16/INT8), pruning and compression, optimized inference frameworks (TensorFlow Lite, OpenVINO, ARM NN), and hardware ac- celerators (Google Coral EdgeTPU, Intel NCS2).

➢ *Object Detection for Edge Environments:*

One of the basic activities of embedded vision is object detection systems and is usually worked out with deep convolutional neural networks (CNNs). There are two major categories of detec- tors: single-stage and two-stage detectors representing various trade-offs between inference speed and detection accuracy.

- *Single-Stage Detectors:*

The localization and classification of single-stage detectors are accomplished in one forward pass. These models are used in real-time edge applications due to low latency.

Representative models include:

✓ *YOLO (You Only Look Once):*

It is created to be used in real time; the lightweight versions of YOLO, namely, YOLOv4-tiny and YOLOv5-nano, can be applied to embedded systems.

✓ *SSD (Single Shot MultiBox Detector):*

It is both fast and accurate with a balance, and lightweight backbones are frequently used.

✓ *MobileNet-SSD:*

Designed to be used in small devices, it comes with a small model size that has satisfactory accuracy.

✓ *EfficientDet-Lite:*

It scales down to mobile/embedded usage with compound scaling and optimized architecture.

The reasons why these detectors are actively used are: low memory footprint, high inference speed, INT8

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

compatibility, and pruning.

- *Two-Stage Detectors:*

Two-stage detectors produce region proposals and then perform further classification and localization, which by definition are usually more accurate but more computationally demanding.

Notable examples include: Faster R-CNN (RPN-based region proposals), Mask R-CNN (segmentation branch), Cascade R-CNN (multi-stage refinement), Sparse R-CNN (learnable proposals), and transformer-based detectors such as DETR and DINO.

Even though these models are capable of being more accurate than lightweight detectors, they are not amenable to real-time deployment on Raspberry Pi without aggressive optimization or external accelerators.

➢ *Object Tracking On Edge Devices:*

Object tracking is the process of identifying the same object and following it over time. Tracking is usually carried out using a live camera feed, video, or a sequence of images captured continuously. It is useful in many embedded applications such as surveillance, traffic monitoring, robotics, autonomous navigation and human-following systems.

In resource-constrained edge devices like Raspberry Pi, tracking becomes challenging because it requires continuous processing. Classical trackers such as KCF and CSRT are still used because they are simple to implement and give fast results. However, these trackers may not perform well when there is occlusion, fast motion, scale changes or illumination variation.

To improve tracking accuracy, multi-object tracking approaches such as SORT and DeepSORT are commonly used. SORT provides tracking based on motion prediction and association, while DeepSORT improves identity tracking by including appearance features. Siamese network based trackers such as SiamFC and SiamRPN are also used for tracking, where the template of the target is matched with the current frame. Recently, tracking pipelines like ByteTrack and BoT- SORT are reported to achieve better association performance, especially in crowded scenes.

Even though deep learning based methods provide better ro- bustness, they increase computational load and memory usage. Running detection and association in every frame can reduce the frame rate in edge devices. Hence, for practical edge implementation, lightweight models and optimization methods are required. Common solutions include reducing detection frequency, using lightweight feature extractors, and applying quantization or other optimization techniques to maintain real- time performance.

IV. VISION ALGORITHMS ON RASPBERRY PI

A. Classical Vision Algorithms On Raspberry Pi

Classical computer vision algorithms continue to be of high application in resource-limited embedded edge devices like the Raspberry Pi because of their low cost and predictable latency, small memory footprint, and their requirement of no GPU acceleration. Such methods are normally applied by applying pipelines based on OpenCV and are based upon deterministic image processing, hand crafted features as well as lightweight tracking mechanisms. The reviewed literature demonstrates that Raspberry Pi devices can achieve practical real-time perception in robotics, surveillance, navigation, and industrial monitoring tasks using purely classical pipelines as well as hybrid classical–deep learning systems.

➢ *Correlation Filter–Based Object Tracking:*

Correlation- filter trackers are widely adopted in Raspberry Pi deployments due to their efficiency. [1] implements a Kernelized Correlation Filter (KCF) tracker on Raspberry Pi 4B for marine object tracking and reports ∼20 FPS at 1280 × 720 resolution using custom real-world marine video. [6] presents a low-cost tracking prototype on Raspberry Pi 3 using KCF, evaluated using publicly available tracking datasets, demonstrating feasibility for economical embedded tracking.

➢ *Feature-Based Detection and Lightweight Embedded Perception:*

Classical feature-based detection pipelines remain common for structured monitoring tasks. [2] employs His- togram of Oriented Gradients (HOG) feature extraction with an SVM classifier for human tracking on Raspberry Pi 3 Model B, validated using custom indoor webcam data. Haar-based detection is also widely used in constrained environments; [12] demonstrates industrial object classification on Raspberry Pi 3B+ using Viola–Jones Haar cascade classifiers, processing a 1.3 MP webcam stream (30 fps input) and evalu- ated using a custom dataset (∼100 images) with a lightweight detector size (< 1 MB).

➢ *Color,Shape and Contour-Based Tracking Pipelines:*

Color segmentation and contour analysis are widely used due to their low computational requirements. [4] implements HSV-based detection and contour-based tracking for real- time ball detection and tracking on Raspberry Pi, validated using experimental trials with colored balls. [17] presents HSV- based object tracking using centroid estimation for robot navigation and obstacle avoidance using Raspberry Pi camera input. [8] demonstrates contour-based detection and centroid displacement tracking for embedded real-time position track- ing using Raspberry Pi camera capture.

➢ *Classical Vision for Structured Navigation Tasks:*

For structured robotic navigation, deterministic processing pipelines provide efficient performance. [3] presents a Rasp- berry Pi-based line follower implemented using grayscale conversion, thresholding, Sobel edge

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

detection, and rule-based motor control, validated through experimental testing using a custom line map. [20] further demonstrates an OpenCV- based line-following vehicle using PID-based control and template matching for reactive behavior on Raspberry Pi 3B+, reporting ∼5.5 FPS at 480 × 320 resolution, validated through experimental evaluation.

- *Surveillance and Embedded Monitoring Using Classical CV:*

Embedded monitoring systems often adopt lightweight vision methods. [13] applies Haar cascade face and eye detection combined with temporal eye-closure logic for driver drowsiness detection on Raspberry Pi. [18] integrates Haar cascade detection with CAMSHIFT tracking for industrial production-line dynamic target tracking on Raspberry Pi. [19] applies segmentation and template matching for banknote authenticity and denomination recognition on Raspberry Pi and reports a detection accuracy of approximately 91.5% on a custom Rupiah banknote dataset. In addition to monitoring and industrial automation, classical tracking pipelines have also been adopted in assistive robotics; [15] demonstrates a Raspberry Pi 3B+ shopping-cart-following robot using CSRT tracking, validated through real-world trials.

- *Algorithmic Enhancements and Classical Tracking Optimization:*

[14] proposes a tracking approach combining Haar detection with Mean-Shift tracking and Locust Search Optimization for victim localization in post-disaster scenarios. The study, which was implemented on Raspberry Pi 3 Model B+, reports a performance of around 1.13 FPS, a memory usage of around 274902 KB, energy efficiency of around 0.226 FPS/W, and a latency of 887.4 ms (Mean-Shift+LSO) relative to 1365.5 ms (Mean-Shift baseline), meaning that performance can be compromised by the inclusion of optimization strategies.

- *Depth-Aware Tracking via Stereo Vision and Filtering:*

Depth-aware tracking has been explored using stereo vision. [16] implements disparity-based tracking using OV7670 stereo cameras combined with ROI-based processing and Kalman filtering on an ESP32 + Raspberry Pi 4B platform, reporting ∼10 FPS, tracking accuracy of approximately 85%, and an error below 10 cm, validated in real-world assistive robot tracking experiments.

- *Hybrid Classical–Deep Learning Pipelines:*

Hybrid ap- proaches are increasingly adopted to improve robustness while reducing computation. [9] combines background subtraction with CNN-based detection for embedded tracking on Rasp- berry Pi. [10] integrates MediaPipe landmark detection with classical image-based visual servoing for robotic arm tracking. [5] integrates SSD-based detection with OpenCV tracking for an intrusion detection and surveillance robot, using a hybrid edge–remote configuration due to compute limitations. [21] proposes a hybrid industrial monitoring pipeline combining pruned YOLOv8 detection with SIFT feature tracking, Lucas–Kanade optical flow, and RANSAC verification, reporting an F1-score ≈ 95%, mAP_{50} ≈ 97%, and tracking accuracy > 95% on a custom industrial bolt dataset.

- *Comparative Benchmarking of OpenCV Trackers:*

Benchmarking studies provide reproducible insights into tracker suitability on embedded devices. [7] evaluates OpenCV trackers (BOOSTING, TLD, MEDIANFLOW, MIL, MOSSE, CSRT, KCF, GOTURN) on Raspberry Pi 3B+, reporting MOSSE at 18.29 FPS, KCF at 7.9 FPS, and CSRT at 3.67 FPS. [11] further benchmarks OpenCV trackers on Raspberry Pi 3B using IoU and success rate metrics, reporting MOSSE at ∼21 FPS, KCF at ∼16 FPS, MEDIANFLOW at 15 FPS, and CSRT at 4 FPS, along with latency values on benchmark sequences such as *Jumping*, *bear-front*, and *z-cup-move1*. The analyzed sources confirm the fact that classical computer vision algorithms are the most appropriate in terms of edge deployment on Raspberry Pi because of their low computational cost, predictable latency, and CPU-only capabilities. Correlation-filter trackers (MOSSE, KCF, and CSRT) offer real-world tracking performance, allowing this to be used in real-time in most robotics and surveillance applications. The feature-based detectors (Haar cascades and HOG-SVM) can be used in controlled conditions, whereas the color-based and contour-based pipelines can be utilized to track a fast object in stable conditions. On the whole, classical methods are an effective starting point of embedded perception especially when the requirements of real-time execution and low-power usage are a priority.

B. Deep Learning Algorithms On Raspberry Pi

The use of deep learning has enhanced object detection performance with hierarchical feature representations that are resistant to illumination change, occlusion, scale variation and background clutter. On edge devices with resource constraints, like the Raspberry Pi, however, the deployment of deep models is still not an easy task because of the constraints on CPU throughput, memory bandwidth and power accessibility. This has led to Raspberry Pi-based deep learning systems that are often based on small-scale architectures, smaller input resolutions, smaller inference times and compression methods like quantization and pruning. As recent literature shows, Raspberry Pi 4 and Raspberry Pi 5 are capable of the practical deep learning inference under these conditions, whereas Rasp- berry Pi 3 is still constrained to run in real-time in CPU-only setups [24], [32].

- *Deep Learning for Object Detection On RASPBERRY PI:*

Object detection is the most widely adopted deep vision task on Raspberry Pi platforms, forming the foun- dation for high-level applications such as surveillance, robotics navigation, industrial inspection, assistive monitoring, and smart transportation. The reviewed works primarily implement deep object detectors using edge-friendly frameworks such as TensorFlow Lite, OpenCV DNN, ONNX Runtime, ARM NN, and OpenVINO, often paired with model compression or hardware acceleration to meet real-time constraints [58]– [63], [65]– [68].

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

- *Single Stage Detector:*

✓ *Lightweight CNN Models for Raspberry Pi–Based Deployment.*

A major category of Raspberry Pi deep vision research emphasizes lightweight CNN architectures designed for embedded inference. Efficient deployment of compact networks such as MobileNet and EfficientNet is demonstrated in industrial inspection and robotics contexts. In [22], MobileNetV3-Large, EfficientNetB3, and ResNet50 are evaluated on Raspberry Pi 5 for real-time detection of hole-type defects on industrial components under CPU-only inference. The study reports ∼8.2 FPS for MobileNetV3 (∼120 ms latency), ∼4.7 FPS for EfficientNetB3 (∼210 ms latency), and ∼3.8 FPS for ResNet50 (∼250 ms latency) using 224 × 224 resized grayscale input. The work also reports model sizes of 13.6 MB (MobileNetV3), 47.0 MB (EfficientNetB3), and 94.7 MB (ResNet50) with parameter counts of 4.2M, 10.7M, and 23.5M, respectively, and is evaluated on 43,482 grayscale industrial images [22]. In robotic deployments, MobileNet- SSD–based human following is implemented on Raspberry Pi 4, validating SSD-MobileNet architectures as feasible detec- tors for embedded navigation tasks [27].

- *Application-Oriented Object Detection Pipelines*

Several works focus on deploying deep learning detection pipelines for practical edge applications such as surveillance, assistive monitoring, and industrial automation. Surveillance- oriented object detection with reduced latency is demonstrated in [23], where MobileNetV2 and EfficientDet-Lite are integrated with entropy-based adaptive buffering. This approach reduces redundant inference and improves responsiveness, confirming that pipeline-level optimizations are essential for CPU-only Raspberry Pi deployments [23]. Assistive technology systems using lightweight YOLO/SSD- class detectors are explored in [25] and [35], demonstrating embedded detection for visu- ally impaired support and safety monitoring [25], [35], [59], [65]. Beyond these, YOLO-driven embedded perception is applied to vision-guided robotic pick- and-place systems on Raspberry Pi, indicating feasibility for industrial manipulation workflows requiring object localization under cost constraints [39]. Collectively, these studies show that Raspberry Pi deep detection pipelines can be applied across varied domains when the model and system design are optimized toward embedded execution [23], [25], [35], [39], [58]– [60], [62], [64]– [66], [68].

- *Benchmarking and Optimization Studies*

Benchmark- driven research provides strong comparative evidence on model suitability for Raspberry Pi [24], [26], [61], [63]. In [24], multiple lightweight detectors including SSD MobileNet V1, SSDLite, and EfficientDet-Lite (Lite0–Lite2) are identified as practical models for Raspberry Pi CPU-only execution, whereas heavier detectors such as YOLOv8 small/medium exhibit high latency and limited real-time feasibility. The same study evaluates Coral USB TPU acceleration across Raspberry Pi 3/4/5 and confirms that hardware acceleration substantially improves FPS and energy efficiency, with minor accuracy degradation due to INT8 quantization. Similarly, [26] provides systematic optimization analysis on Raspberry Pi platforms using INT8-quantized models at reduced reso- lutions. On Raspberry Pi 4 (320×320 input), it is reported in the study that the FPS was about 8 (SSD- MobileNet- V1 FPN). Approx. 12.5 FPS (SSD-MobileNet-V2 FPNLite), 12 FPS (EfficientDet-Lite0), and 33 FPS (YOLOv5n), and approximate reported latency of about 125 ms, 80 ms, 83 ms, and 30 ms, respectively, on MS COCO 2017 at INT8 quantization on TensorFlow Lite [26]. These benchmarking works define that model-hardware co-design is necessary to detect embedded objects on Raspberry Pi [24], [26].

- *YOLO Deployment on Raspberry Pi Platforms*

YOLO- family models remain widely adopted for Raspberry Pi object detection due to their high accuracy and strong community tooling; however, feasibility depends on selecting lightweight variants and applying quantization or optimized inference [58], [60], [62], [64], [66], [68]. YOLOv7 and YOLOv7-tiny are evaluated on Raspberry Pi 4 for traffic monitoring in [31], confirming deployment feasibility under optimized inference but also highlighting performance limitations in CPU-only settings for larger models. Raspberry Pi 5 benchmarking of YOLOv8n in drone-oriented constrained edge scenarios fur- ther confirms that nano-scale YOLO models can sustain real- time-like throughput under CPU-only execution [30]. High- speed deployment High-speed deployment is demonstrated in [28] where quantized YOLOv4-Tiny is deployed on Raspberry Pi 5 using post-training quantization of TensorFlow Lite and 45.1 FPS at 416x416 input resolution is reported, with latency of 183 ms and model-size of 6.4 MB. Power consumption has also been mentioned in the study as about 4 W when deployed [28]. The prototype implementations also point to using YOLOv8 pipelines to detect and recognize people on Raspberry Pi 4B, which suggests an increasing interest in detecting and providing the edge device with higher-level analytics [34].

- *Hybrid and Comparative Deep Detection Studies*

Combined strategies and comparative analysis give information on the trade offs when deploying Raspberry Pi. MobileNet-SSD inference and embedded deployment in foggy traffic condi- tions are achieved in [29] with the help of Raspberry Pi 4 and Coral TPU, which enable robust detection-based monitoring with tracking-based quality metrics. Multi- generation YOLO comparisons on Raspberry Pi 4B in [38] show that tiny-scale models provide the most practical embedded trade-off, while heavier YOLO variants impose high CPU usage and execution time that may block concurrent robotic processing. In addi- tion, hybrid stabilization strategies combining YOLO detection with estimation filtering are explored in [37], indicating that integrating temporal filtering can improve detection stability, although the study is evaluated in simulation rather than direct Raspberry Pi real- time deployment.

- *Limitations on Raspberry Pi 3 and Low-End Edge Devices*

Raspberry Pi 3 devices remain notably constrained for deep learning object detection, particularly without accelera- tion. SSD-based human detection for ADAS-style monitoring

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

on Raspberry Pi 3 Model B+ achieves very low throughput (∼0.8 FPS), showing that CPU-only Pi 3 is not suitable for real-time detection systems requiring fast response [32]. The surveillance robotics implementation running YOLOv3-Tiny on Raspberry Pi 3B+ indicates viability only at relatively low FPS and in simple scenes due to resource limitations [36]. These results support the conclusion that hardware acceleration or ultra-light models must be used to deploy Pi 3 generation implementations [63], [67].

- *Deployment Toolchains and Embedded AI Platforms*

Another crucial enabler of Raspberry Pi deep learning adop- tion is also mentioned as deployment toolchains. Embedded ML platforms like Edge Impulse can support the real-world implementation of Tiny-YOLO models on Raspberry Pi 4B with less integration work, and it can be used to perform quick prototyping and deploy deep detectors to edges [33]. These toolchains work hand-in-hand with performance optimization, to make the model conversion, quantization, and on-device testing processes simpler.

- *Efficiency-Oriented Model Redesign*

Efficiency-driven model redesign remains a promising direction to improve Raspberry Pi inference feasibility. In [40] , an efficient method (YOLIC) is proposed for object localization and classification on edge devices, reporting real-time performance on Rasp- berry Pi 4B by integrating lightweight layers and pruning. Such architectural simplification demonstrates that embedded- focused design can achieve improved throughput while retain- ing usable detection performance, making it suitable for CPU- only Raspberry Pi deployments [40].

The reviewed deep learning detection studies con- firm that Raspberry Pi devices can support practical edge inference only when lightweight CNN architectures (MobileNet/SSD/EfficientDet/YOLO-tiny/nano) and system- level optimizations are adopted. Benchmark and optimization works consistently show that input resolution, backend/runtime (TensorFlow Lite/ARM NN/OpenVINO/ONNX), and quanti- zation (INT8) are critical factors affecting FPS and latency [24] , [26] , [61] , [63] . Raspberry Pi 5 significantly im- proves feasibility of real-time detection compared to Raspberry Pi 3/4, but heavier YOLO variants still remain constrained under CPU-only execution. Overall, the literature establishes that model–hardware co- design and compression/acceleration strategies are essential to enable scalable real-time detection applications on Raspberry Pi platforms.

✓ *Two Stage Detector:*

Deep learning–based vision deployment on Raspberry Pi has been widely explored for surveillance-style monitoring tasks such as object counting, attendance monitoring, face-based access control, and IoT-driven alerting applications. These works typically adopt either lightweight single-stage detectors (YOLO variants, SSD-MobileNet) for speed or two- stage detectors (Faster R-CNN) for higher accuracy, demon- strating the core trade-off between throughput and detection robustness in resource-constrained inference environments.

Two-stage detection frameworks are particularly used when higher recognition accuracy is required. In [9] , Faster R- CNN (Inception V2 backbone) and SSD MobileNet V2 are compared for on-device object detection and counting on Raspberry Pi 4 (8 GB), where Faster R-CNN achieves around 9 FPS while SSD reaches approximately 22 FPS at 640×480 resolution. The work reports detection quality through preci-sion/recall and IoU and shows that SSD-based pipelines are more suitable for real-time on-device counting, whereas Faster R-CNN offers comparatively stronger detection performance at higher latency. In the same way, Faster R-CNN can also be used in the case of smart building automation. A Faster R- CNN (VGG-16 based backbone) face recognition pipeline is built into an IoT controlled door access automation, cloud log- ging, and messaging service-based alerting in [42] , which high classification performance with an accuracy of about 99.3% on a custom dataset of employees. Although the frame rate in real time is not specifically stated, the work demonstrates the viability of using deep vision together with embedded IoT control services in Raspberry Pi.

Lightweight detectors are highly preferred by human counting and monitoring applications due to their acceptable real time performance. Student counting within the laboratory is presented in [42] with the help of YOLOv3-Tiny and Faster R-CNN ResNet50 on Raspberry Pi 3B+. The findings reveal that YOLOv3-Tiny demonstrates between 4.17 and 4.89 FPS, and Faster R-CNN performance is lower than 1 FPS, and therefore, it can be concluded that heavy detectors cannot be used in continuous monitoring on Raspberry Pi 3-class hardware. It also incorporates entry/exit line logic and ID-based association in the work, which proves that it is possible to implement detection-based counting on Raspberry Pi in case of lightweight models chosen.

Accelerators have also been used to improve edge-assisted monitoring applications. A Faster R-CNN-based indoor safety monitoring system is introduced in [43] based on Raspberry Pi 4 and an Intel Neural Compute Stick (Movidius) to allow low- cost deployment to monitor indoor capacity, social distancing and mask detection. The system considers the accuracy of the results, recall, F1-score, and the custom orientation-related information (AP/AOS/OS), the F1 score being stated as 1.00 and the accuracy as high as approximately 0.9948. Although real-time FPS is not explicitly specified, the integration of an accelerator highlights an important direction for enabling complex two-stage networks on Raspberry Pi systems.

Overall, these studies confirm that deep learning deploy- ment on Raspberry Pi is feasible for real-world monitoring applications when detector selection is aligned with hardware limits. SSD-MobileNet and YOLO-tiny/nano variants consis- tently provide practical throughput for real-time monitor- ing, while Faster R-CNN pipelines remain better suited for accuracy-critical scenarios and typically require either higher- end Raspberry Pi configurations or accelerator support to become practical in continuous operation environments.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

➢ *Deep Learning for Object Tracking On Raspberry Pi:*

Deep learning–based tracking on Raspberry Pi has gained strong research interest due to the increasing demand for real- time multi-object perception on low-cost embedded systems under strict constraints such as limited CPU compute capability, restricted memory, and low power availability. Recent literature shows that Raspberry Pi plat- forms (Pi 3B+, Pi 4B, Pi 5) can support practical detection- and-tracking pipelines when lightweight detectors (YOLO tiny/nano, MobileNet-SSD, EfficientDet-Lite) are combined with computationally efficient tracking algorithms such as DeepSORT, ByteTrack, and OC-SORT. In addition, several recent papers emphasize hybrid strategies, where classical vision or motion processing is fused with deep learning to reduce deep inference load and improve real-time feasibility on CPU-only embedded platforms [69] , [72] , [73] .

- *Multi-Object Detection + DeepSORT Tracking Pipelines*

A major portion of Raspberry Pi tracking literature uses a pipeline combining YOLO-based object detection with DeepSORT-based association to achieve identity-consistent multi-object tracking. In [44], YOLOv8n with DeepSORT is integrated with RFID sensing for enhanced human tracking on Raspberry Pi 3B+, reporting high detection precision and practical real-time performance. Similarly, [46] demonstrates acceleration of recognition and tracking using Intel NCS2 (Myriad X) with Raspberry Pi 4B, showing that external accelerators can improve throughput and FPS-per-watt per- formance in DeepSORT-based tracking pipelines. Multi- object tracking on Raspberry Pi is also demonstrated using YOLOv4- tiny + DeepSORT in [52], where CPU-only deployment on Raspberry Pi 4B achieves approximately 3–5 FPS with la-tency around 200–250 ms at 416 × 416 resolution, reporting MOTA ≈ 0.71 and mAP around 38–40% on a custom indoor surveillance dataset.

- *Detection-Only Tracking (Per-frame Following Applications)*

Some Raspberry Pi tracking applications rely on per-frame detection without explicit multi-object tracking algorithms. In [45] , a cost-effective person-following system is implemented using a retrained YOLOv3-tiny model, where navigation deci- sions are derived directly from detection outputs. Such systems confirm that in assistive robotics applications, robust per-frame detection can be sufficient when motion conditions are con- trolled. However, identity persistence and occlusion robustness remain limited compared to association-based tracking.

- *Classical Tracking Enhanced by Deep Detection (Centroid / Servo / PID Tracking)*

For lightweight robotic tracking tasks, detection is often combined with simple tracking logic such as centroid dis-placement, servo-based following, or PID control. In [47] , YOLOv3-Tiny detection is paired with centroid tracking on an educational Raspberry Pi robot for ball tracking, reporting ap- proximately 6–10 FPS at 416×416 resolution, latency around 100–150 ms, and detection accuracy above 90%. Similarly,[51] applies YOLOv5-Lite with PID-based servo tracking for embedded beehive detection on Raspberry Pi 4B,reporting 3.15 FPS with 317 ms latency at 640×640 resolution, model size of 2.97 MB, and 0.78M parameters, with 99.50% mAP on a custom beehive dataset. In [70] , a Raspberry Pi 4–based farm monitoring system integrates deep learning models such as YOLOv8n, MobileNetSSD, and EfficientNet-B0 to identify pigs and monitor motion, where centroid-based tracking and Euclidean distance calculations provide movement estimation and visualization.

- *ByteTrack / OC-SORT and Modern MOT Pipelines on Raspberry Pi)*

Modern MOT algorithms such as ByteTrack and OC-SORT are increasingly explored due to strong tracking performance with reduced reliance on heavy appearance embeddings. In [55] , underwater multi-object tracking is demonstrated on Raspberry Pi 4 using YOLOv5n/YOLOv8n combined with ByteTrack and OC-SORT, reporting real-time feasibility with strong MOTA and IDF1 results. Comparative evaluation in [56] further highlights performance differences between detec- tor selection (YOLOv5n vs YOLOv8n) and tracker selection (ByteTrack vs OC-SORT), confirming that lightweight asso- ciation methods improve feasibility for embedded Raspberry Pi deployments.

- *Accelerator-Based Edge Tracking (Coral / Movidius/ OpenVINO)*

Several papers demonstrate that Raspberry Pi can achieve improved tracking performance when paired with accelera- tors. In [57] , SSD-MobileNet v2 + DeepSORT is deployed on Raspberry Pi 3B using Intel Movidius Neural Compute Stick for UAV tracking, reporting 8.3 FPS with 120 ms latency, detection performance of 67.5% mAP on UAVDT, and tracking performance of MOTA = 81.6 and MOTP = 79.2, with reported power consumption of 15.3 W. In addition, [53] presents an energy-efficient tracking system deployed on Raspberry Pi 4B + Coral EdgeTPU, evaluating detectors such as SSD-MobileNet V1/V2, EfficientDet-Lite0, and YOLOv5 variants under a DeepSORT-like tracking pipeline. The study reports reduced frame-to-frame latency values (e.g., 104 ms for SSD-MobileNet V1) evaluated on MOT15 sequences and power consumption of approximately 5.4 W under CPU-only operation and 6.3 W with EdgeTPU acceleration.

- *Alternative Tracking Approaches Beyond Deep Association Trackers)*

Not all Raspberry Pi tracking works use association-based MOT tracking. In [49] , a particle filter–based tracking scheme is implemented on Raspberry Pi 3 Model B with target remodeling and reinitialization, reporting approximately 13.5 FPS at 320×240 resolution, evaluated on custom test videos and PETS2000, with reported power consumption of approximately 3.2 W. Hybrid robotics perception pipelines are also explored. In [50] , monocular depth estimation combined with visual odometry supports pose tracking and navigation tracking on low-resource Raspberry Pi-class platforms. In addition, [72] proposes hybrid tracking methods initialized using YOLOv5 detections and enhanced using Kalman Filter integration with KCF and TLD tracking components, improv- ing robustness

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

for occlusion and deformation under search- and-rescue conditions.

- *Emerging Raspberry Pi 5 Deep Tracking and Safety Monitoring Systems)*

The Raspberry Pi 5 generation enables more advanced edge deployments due to improved CPU performance. In [48], an improved YOLO framework is deployed for escalator safety monitoring on Raspberry Pi 5, reporting 7.74 FPS under Open- VINO deployment at 640×640 resolution with latency of 129.2 ms, demonstrating improved feasibility for intelligent monitor- ing tasks. Furthermore, multi-algorithm fusion approaches are emerging. In [74], a face tracking and recognition framework is designed for Raspberry Pi– controlled robots by integrating MTCNN for face detection, SORT for identity assignment and tracking, and Inception- ResNet for face recognition, achieving over 99% recognition accuracy on the LFW dataset with optimized similarity thresholds and real-time feasibility through multi-threading and code-level refinements.

- *Hybrid Classical–Deep Tracking for Efficient Em- bedded Deployment)*

Recent work increasingly highlights hybrid strategies that reduce deep inference frequency by combining classical vi- sion detection and deep learning refinement. In [69], vehicle tracking is performed using background subtraction as motion-based detection combined with selective YOLO-based CNN detection. The system eliminates unnecessary CNN inference calls and achieves a matching rate above 90% with average speed around 16 FPS on Raspberry Pi 4. Similarly, [73] proposes a lightweight two-step ball detection approach for soccer-playing robots where coarse region extraction is per- formed using integral image–based feature extraction and can- didate ROIs are refined using a lightweight CNN. The method achieves 97.17% detection accuracy and further improves ro- bustness using spatio-temporal correlation of historical frames under blur and partial occlusion.

The reviewed deep learning–based tracking literature con- firms that Raspberry Pi platforms can support real-time or near real-time tracking when lightweight detection backbones are selected (YOLO nano/tiny, MobileNet-SSD, EfficientDet- Lite), tracking strategies are chosen based on embedded feasibility, and deployment toolchains and accelerators are leveraged when required. Raspberry Pi 3B+ remains limited for scalable multi-object tracking without acceleration, while Raspberry Pi 4B and especially Raspberry Pi 5 demonstrate strong potential for practical multi-object tracking deploy- ments across robotics, surveillance, agriculture, and industrial monitoring applications.

V. CONCLUSION

The current review was a unified literature review on visual intelligence algorithms that run on resource-limited edge devices powered by Raspberry Pi and with a focus on object detection and multi-object tracking pipelines. Literature survey supports the idea that Raspberry Pi platforms have the potential to support useful embedded vision applications such as surveillance, mobile robotics, assistive monitoring, agricul- ture/livestock monitoring and intelligent transportation, under the strict constraints of CPU throughput, memory footprint, inference latency, and power consumption.

The classical computer vision techniques are still very applicable to the Raspberry Pi applications because of low computational cost, predictability of the execution time, and applicability to structured environment. Background subtrac-tion techniques, contour-based segmentation, optical flow, and correlation filter trackers (KCF, MOSSE, CSRT) remain an important part of embedded pipelines and deliver consistent real-time performance in lightweight tracking scenarios par- ticularly where the scene is controlled and the difference in appearance is not high.

In case of deep learning based perception, it is al-ways emphasized that lightweight single stage detectors like MobileNet-SSD, EfficientDet-Lite and YOLO tiny/nano ver-sions offer the most viable trade-off between accuracy and speed in the embedded inference via CPU only. Heavier models, on the other hand, usually cause a high latency and power cost, making them not configurable to do continuous real- time processing, unless highly optimized or accelerated. Consequently, the ability to deploy could greatly be influenced by optimization options such as resolution-reduction, model- pruning, INT8-quantization, and the adoption of efficient inference-runtimes such as TensorFlow Lite, OpenVINO, and ARM NN.

Tracking studies also prove that a Raspberry Pi can be used to enable multi-object tracking in cases where lightweight detection backbones are used to run computationally effi- cient trackers. Association algorithms like ByteTrack and OC- SORT are becoming popular to be deployed on a modern embedded platform, because they possess a high identity consistency with the advantage of not incurring an intensive computational cost on appearance embedding extraction. Com- paratively, pipelines based on DeepSORT in general need a lot of optimization or accelerators to achieve real-time per- formance. Moreover, some hybrid approaches where classical motion-based detectors or region proposals are combined with selective deep inferences are also a promising way of attaining real-time tracking on low-power devices.

On the whole, this review has found that Raspberry Pi can be a viable platform to implement edge-based visual intelligence when (i) lightweight models, (ii) model-software optimization of detection-tracking pipelines, and (iii) when more throughput is needed, the Google coral EdgeTPU or Intel Neural Compute Stick(s) along with an SDL are used as accelerators. Future studies can be used to devise embedded-first detection-tracking models that can so-jointly trade accu-racy, computation and energy consumption, allowing scalable real-time implementation over next generation Raspberry Pi platforms.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

REFERENCES

[1] V.N.Le,V.L.Khong,V.T.Nguyen,and Q. D. Pham, "Development of a real-time object-tracking system using Raspberry Pi," JMST, vol. 90, no. 90, pp. 127–133, Oct. 2023.

[2] L. Rosyidi, A. Prasetyo, and M. S. Romadhon, "Object tracking with Raspberry Pi using Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM)," in Proc. 8th Int. Conf. Inf. Com- mun. Technol. (ICoICT), Yogyakarta, Indonesia, 2020, pp. 1–6, doi: 10.1109/ICoICT49345.2020.9166330.

[3] G. Dewantoro, J. Mansuri, and F. D. Setiaji, "Comparative study of computer vision based line followers using Raspberry Pi and Jetson Nano," Jurnal Rekayasa Elektrika, vol. 17, no. 4, pp. 239–246, Dec. 2021, doi: 10.17529/jre.v17i4.21324

[4] Aryani, K. Dewi, F. TaBy, and E. P. Sanggaria, "Real-time ball detection and tracking using Raspberry Pi," INTEK Jurnal Penelitian, vol. 10, no. 1, pp. 48–54, 2023, doi: 10.31963/intek.v10i1.4301.

[5] S. Sheikh, P. Kumar, R. Kshirsagar, M. Chouhan, and A. S. Bhagwat, "Raspberry Pi based surveillance robot for real-time intrusion detection and tracking," Int. Res. J. Eng. Technol. (IRJET), vol. 8, no. 5, pp. 3333–3335, May 2021.

[6] T. J. Nagalakshmi and P. Prakas, "Fast and economical object tracking using Raspberry Pi 3.0," Int. J. Eng. Adv. Technol. (IJEAT), vol. 8, no. 6S, Aug. 2019, doi: 10.35940/ijeat.F1070.0886S19

[7] R. R. Patil, O. S. Vaidya, G. M. Phade, and S. T. Gandhe, "Qualified scrutiny for real-time object tracking framework," Int. J. Emerging Technol., vol. 11, no. 3, pp. 313–319, 2020.

[8] G. Anand and A. K. Kumawat, "Object detection and position tracking in real time using Raspberry Pi," Materials Today: Proceedings, vol. 47, pp. 3221–3226, Jul. 2021, doi: 10.1016/j.matpr.2021.05.276.

[9] M. Ojha, S. Prajapati, and M. Jain, "Object detection and tracking on a Raspberry Pi using background subtraction and CNNs," Int. J. Comput. Appl., vol. 182, no. 47, pp. 6–10, Feb. 2019.

[10] U. K. Sahu, K. M., K. A., M. P. M., A. Jaiswal, U. K. Yadav, and S. K. Dash, "Autonomous object tracking with vision-based control using a 2DOF robotic arm," Scientific Reports, vol. 15, no. 1, 2025, doi: 10.1038/s41598-025-97930-3.

[11] "Comparison of API trackers in OpenCV using Raspberry Pi hard- ware," Science Arena Publications, vol. 6, no. 2, pp. 1–9, 2020, doi: 10.51847/fo5kgrb7zp.

[12] T. Serrano-Ram´ırez, N. D. C. Lozano-Rinco´n, A. Mandujano-Nava, and Y. J. Sa´mano-Flores, "Artificial vision system for object classification in real time using Raspberry Pi and a web camera," Revista de Tec- nolog´ıas de la Informacio´n y Comunicaciones, pp. 20–25, 2021, doi: 10.35429/jitc.2021.13.5.20.25.

[13] P. Mohadese et al., "Multi-modal driver drowsiness detection in ADAS via attention-guided Siamese network with temporal modeling," Int. J. Web Res., vol. 9, no. 1, pp. 40–55, 2026.

[14] R. Hardini and Y. Bandung, "Design and implementation of object tracking system based mean-shift with locust search optimization on Raspberry Pi," in Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI), 2020, pp. 282–287, doi: 10.1109/ICITSI50517.2020.9264941.

[15] N. D. Perez, J. Villaverde, W. Cereneo, A. Custodio, S. B. Conde, and L. Bulawan, "Design and development of a Raspberry Pi-based shopping cart following robot through computer vision and object tracking method," in Proc. IEEE Region 10 Symp. (TENSYMP), 2022, pp. 1–6, doi: 10.1109/TENSYMP54529.2022.9864520.

[16] F. Mart´ınez, F. Mart´ınez, and C. Penagos, "Integrating low-cost vision for autonomous tracking in assistive robots," Bull. Electr. Eng. Inform., vol. 14, no. 3, pp. 1881–1889, 2025, doi: 10.11591/eei.v14i3.9242.

[17] G.-C. Cre¸t, L. T¸ epelea, C. Grava, I. Gavrilu¸t, and T. Suru, "Using Raspberry Pi and the OpenCV library to command a robot for tracking a colored object," in Proc. 17th Int. Conf. Eng. Modern Electr. Syst. (EMES), 2023, pp. 1–4, doi: 10.1109/EMES58375.2023.10171765

[18] X. Tian, H. Feng, and J. Chen, "An industrial production line dynamic target tracking system based on HAAR and CAMSHIFT," School of Electromechanical Engineering, Lingnan Normal University, Zhanjiang, China, Mar. 13, 2020.

[19] S. Gupta, R. Gupta, S. Shah, J. Chauhan, A. Pandey, and G. Ramasamy, "A Currency and Denomination Detection System using Raspberry Pi and Machine Learning," 2025, doi: 10.4108/eai.28-4-2025.2357994.

[20] J. Li, "OpenCV-based PID control line following vehicle with object recognition and reaction," in AIP Conference Proceedings, vol. 3144, no. 1, Art. no. 050009, Jun. 26, 2024, doi: 10.1063/5.0214211.

[21] P. Jiang, Y. Geng, Z. Sang, and C. Wang, "Real-time bolt loosening monitoring method based on edge device deployment and optical flow tracking," J. Failure Anal. Prevent., vol. 25, no. 4, Jul. 2025, doi: 10.1007/s11668-025-02186-8.

[22] M. Deniz, I. Bogrekci, and P. Demircioglu, "Real-time detection of hole- type defects on industrial components using Raspberry Pi 5," Applied System Innovation, vol. 8, no. 4, p. 89, 2025.

[23] P. Chandrashekar and P. M. Sajjanar, "Real-time, low-latency surveil- lance using entropy-based adaptive buffering and MobileNetV2 on edge devices," arXiv preprint, arXiv:2506.14833v1, 2025.

[24] K. Alqahtani, M. A. Cheema, and A. N. Toosi, "Benchmarking deep learning models for object detection on edge computing devices," in Lecture Notes in Computer Science, pp. 142–150. Springer, 2024, doi: 10.1007/978-981-96-0805-8-11.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

[25] K. Manjari, M. Verma, G. Singal, and V. Chamola, "Catalysing assis- tive solutions by deploying light-weight deep learning model on edge devices," J. Exp. Theor. Artif. Intell., vol. 37, no. 3, pp. 465–486, 2023, doi: 10.1080/0952813X.2023.2219286.

[26] Myroniuk and B. Blagitko, "Optimizations of deep learning ob- ject detection models for inference acceleration on general-purpose and hardware-accelerated single-board platforms," Electronics and In- formation Technology, vol. 29, no. 6, pp. 57–68, Apr. 2025, doi: 10.30970/eli.29.6.

[27] V. Kamath and R. Adige, "Investigation of MobileNet-SSD on human follower robot for stand-alone object detection and tracking using Raspberry Pi," Cogent Engineering, vol. 11, no. 1, Apr. 2024, doi: 10.1080/23311916.2024.2333208.

[28] S. Boddu and A. Mukherjee, "Efficient edge deployment of quantized YOLOv4-Tiny for aerial emergency object detection on Raspberry Pi 5," arXiv preprint, arXiv:2506.09300, 2025.

[29] J. Park, J. Hong, W. Shim, and D. Jung, "Multi-object tracking on SWIR images for city surveillance in an edge-computing environment," Sensors, vol. 23, no. 14, p. 6373, 2023, doi: 10.3390/s23146373.

[30] L. Rey, A. M. Bernardos, A. D. Dobrzycki, D. Carramin˜ana, L. Bergesio, J. A. Besada, and J. R. Casar, "A performance analysis of you only look once models for deployment on constrained computational edge devices in drone applications," Electronics, vol. 14, no. 3, p. 638, 2025, doi: 10.3390/electronics14030638.

[31] J. C. da Silva, M. C. Silva, E. J. Luz, S. Delabrida, and R. A. Oliveira, "Real-time object detection performance analysis using YOLOv7 on edge devices," IEEE Latin America Transactions, 2023.

[32] Mulyanto, R. I. Borman, P. Prasetyawan, W. Jatmiko, and P. Mur- santo, "Real-time human detection and tracking using two sequential frames for advanced driver assistance system," in Proc. 3rd Int. Conf. Informatics and Computational Sciences (ICICoS), 2019, pp. 1–5, doi: 10.1109/ICICoS48119.2019.8982396.

[33] N. A. Gookyi, F. A. Wulnye, M. Wilson, P. Danquah, S. A. Danso, and A. A. Gariba, "Enabling intelligence on the edge: Leveraging Edge Impulse to deploy multiple deep learning models on edge devices for tomato leaf disease detection," AgriEngineering, vol. 6, no. 4, pp. 3563–3585, 2024.

[34] P. Kumar, S. K. Choudhary, S. Pallerla, S. Sangani, A. Thota, and V. Matta, "Pi-Vision: Leveraging YOLO v8 for real-time person detection and recognition on Raspberry Pi," in Proc. 5th Int. Conf. Data Intelli- gence and Cognitive Informatics (ICDICI), 2024, pp. 1446–1450, doi: 10.1109/ICDICI62993.2024.10810836

[35] S. S., P. Dr., R. Manjunatha, R. Stephen, S. Melwin, and V. Prasad, "Real-time object detection system using YOLO and Raspberry Pi," Int. J. Innovative Research in Technology, vol. 11, no. 12, pp. 5192–5193, 2025.

[36] P. Thumati, S. J. R., V. P., and R. Karipe, "An affordable surveillance robot using Raspberry Pi and YOLO," Int. J. Engineering Research and Technology, vol. 14, no. 4, p. 275, 2025.

[37] Moksyakov, Y. Wu, S. A. Gadsden, J. Yawney, and M. AlShabi, "Object detection and tracking with YOLO and the sliding innovation filter," Sensors, vol. 24, no. 7, p. 2107, 2024, doi: 10.3390/s24072107.

[38] S. Cherubin, "YOLO object detection and classification using low-cost mobile robot," Przeglad Elektrotechniczny, vol. 1, no. 9, pp. 31–35, 2024, doi: 10.15199/48.2024.09.04.

[39] H. Kumar, D. R. K. Raja, S. Suresh, R. Kottamala, and M. Harsith, "Vision-guided pick and place systems using Raspberry Pi and YOLO," in Proc. IEEE Int. Conf. Next-Generation Electronic Systems (IC- NEWS), 2024, pp. 1–7, doi: 10.1109/ICNEWS60873.2024.10731108.

[40] K. Su, Y. Tomioka, Q. Zhao, and Y. Liu, "YOLIC: An efficient method for object localization and classification on edge devices," Image and Vision Computing, vol. 147, p. 105095, 2024, doi: 10.1016/j.imavis.2024.105095.

[41] N. N. F. Taek and N. A. Sony, "Design and implementation of a student counting and monitoring system in a laboratory using human tracking method with OpenCV and TensorFlow," J. Robot., Autom. Electron. Eng., vol. 2, no. 1, pp. 20–29, 2024, doi: 10.21831/jraee.v2i1.554.

[42] Al-Absi, H. A. Al-Mashhadani, S. A. Mostafa, and M. A. Mo- hammed, "Smart office automation system based on Raspberry Pi and Faster R-CNN for face recognition," Array, vol. 17, p. 100268, 2023.

[43] J. Vega, "Control system for indoor safety measures using a Faster R- CNN architecture," Electronics, vol. 12, no. 11, Art. no. 2378, 2023, doi: 10.3390/electronics12112378.

[44] A. Hamad, T. A. Salih, and A. F. Mahmoud, "Integration of DeepSORT and RFID technology for enhanced human tracking," NTU Journal of Engineering and Technology, vol. 3, no. 4, pp. 17–25, 2024, doi: 10.56286/ntujet.v3i4.1095.

[45] Boschi, F. Salvetti, V. Mazzia, and M. Chiaberge, "A cost-effective person-following system for assistive unmanned vehicles with deep learning at the edge," Machines, vol. 8, no. 3, p. 49, 2020, doi: 10.3390/machines8030049.

[46] T. Gao and J. Suto, "Acceleration of image classification and object tracking by the Intel Neural Compute Stick 2 with power efficiency evaluation on Raspberry Pi 4B," Sensors, vol. 25, no. 6, p. 1794, Mar. 2025, doi: 10.3390/s25061794.

[47] M. Sikora, S. R. Chirumamilla, T. Rojek, and J. Siemin´ski, "Ball detection using deep learning implemented on an educational robot based on Raspberry Pi," Sensors, vol. 23, no. 8, p. 4071, Apr. 2023, doi: 10.3390/s23084071

[48] Q. Wang, D. Wang, J. Lu, G. Xiao, D. Liang, G. Lu,

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

and H. Shao, “SAL- YOLO-DeepSeek: A lightweight real-time detection and LLM-driven decision framework for intelligent escalator safety monitoring,” Scien- tific Reports, vol. 15, Art. no. 40600, Nov. 2025, doi: 10.1038/s41598- 025-24260-9.

[49] J. Panda, P. K. Nanda, and T. Pradhan, “Particle filter-based video object tracking scheme with target remodeling and reinitialization and its hardware implementation using Raspberry Pi,” IEEE Access, vol. 12, pp. 98285–98305, 2024, doi: 10.1109/ACCESS.2024.3428321.

[50] Vashisht, G. C. Gandhi, S. Kalra, and D. K. Saini, “Hybrid robot navigation: Integrating monocular depth estimation and visual odom- etry for efficient navigation on low-resource hardware,” Computers and Electrical Engineering, vol. 124, Art. no. 110375, 2025, doi: 10.1016/j.compeleceng.2025.110375.

[51] P. Gao et al., “Dynamic beehive detection and tracking system based on YOLO v5 and unmanned aerial vehicle,” Journal of Biosystems Engineering, vol. 47, no. 4, pp. 510–520, 2022, doi: 10.1007/s42853-022-00166-6.

[52] M. Danish, R. Verma, J. Brazauskas, I. Lewis, and R. Mortier, “Deep- Dish: Multi-object tracking with an off-the-shelf Raspberry Pi,” in Proc. 3rd ACM Int. Workshop on Edge Systems, Analytics and Networking (EdgeSys), 2020, pp. 37–42, doi: 10.1145/3378679.3394535.

[53] M. Danish, R. Verma, J. Brazauskas, I. Lewis, and R. Mortier, “Deep- Dish on a diet: Low-latency, energy-efficient object detection and track- ing at the edge,” in Proc. 17th European Conf. on Computer Systems (EuroSys), 2022, pp. 43–48, doi: 10.1145/3517206.3526273.

[54] J. Brazauskas, C. Jensen, M. Danish, I. Lewis, and R. Mortier, “Cer- berus: Privacy-preserving crowd counting and localisation using face detection in edge devices,” in Proc. 19th European Conf. on Computer Systems (EuroSys), 2024, pp. 25–30, doi: 10.1145/3642968.3654817.

[55] R. ElTobgui, “Visual perception of underwater robotic swarms,” Master’s thesis, 2024.

[56] N. M. Jain and N. A. Shah, “Convolutional neural networks for real- time object detection with Raspberry Pi,” World Journal of Advanced Engineering Technology and Science, vol. 4, no. 1, pp. 87–105, 2021, doi: 10.30574/wjaets.2021.4.1.0067.

[57] S. Hossain and D. Lee, “Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU- based embedded devices,” Sensors, vol. 19, no. 15, p. 3371, Aug. 2019, doi: 10.3390/s19153371.

[58] N. Rathour et al., ”Advanced Security with YOLO Object Detection Using Raspberry Pi,” 2024 International Conference on Emerging Tech- nologies and Innovation for Sustainability (EmergIN), Greater Noida, India, 2024, pp. 214-218, doi: 10.1109/EmergIN63207.2024.10960931.

[59] S. N. Katkade, R. R. Manza, and C. Pattebahadur, “YOLOv5-Based Object Detection System for Visually Impaired Individuals Using Rasp- berry Pi,” Artificial Intelligence and Applications, vol. XX, no. XX, pp. 1–5, Jul. 2025, doi: 10.47852/bonviewAIA52024434.

[60] W. Nugroho, R. Zahabiyah, M. J. Arifiant, and A. Afianto, “Automated Component Detection for Quality PCB Using YOLO Algorithm with IoT Real-Time Streaming on Raspberry Pi”, INFOTEL, vol. 17, no. 2, pp. 440–455, Jul. 2025.

[61] Pranav Krishnan, Sahithya Kattamuri, Gayathri R Prabhu, and Manazhy Rashmi. 2024. Assistive Eye: A Comparative Analysis of YOLO Ob- ject Detection Models on Edge Devices. In Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing (IC3- 2024). Association for Computing Machinery, New York, NY, USA, 104–108. https://doi.org/10.1145/3675888.3676037

[62] Yuvaraj R, Senthil Kumar D, Sunil Arjun Bhalerao, Krishnan Muruge- san, Suresh Vellaiyan, Nguyen Van Minh,“Real-time fire detection and suppression system using YOLO11n and Raspberry Pi for thermal safety applications”,Case Studies in Thermal Engineering,

[63] V. L. B. Silva, F. A. P. de Figueiredo and S. B. Mafra, ”Performance Evaluation of Edge Computing Object Detection Models for Maritime Surveillance on a Raspberry Pi,” 2024 IEEE Latin-American Conference on Communications (LATINCOM), Medellin, Colombia, 2024, pp. 1-6, doi: 10.1109/LATINCOM62985.2024.10770682.

[64] Y. Setiawan, M. N. Puji and W. Astuti, ”Traffic Signs Detection System Using YOLO (You Only Look Once) That Provides Notifi- cation,” 2024 IEEE International Conference on Smart Mechatronics (ICSMech), Yogyakarta, Indonesia, 2024, pp. 95-100, doi: 10.1109/IC-SMech62936.2024.10812285.

[65] K. Hari, M. A. Chowdary, M. Sumathi, D. Sainadh and T. Manikanta, ”Deployment of Real-Time Object Recognition in Rasp- berry Pi with Neural Compute Stick for Blind and Deaf People,” 2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 2024, pp. 305-310, doi: 10.1109/ICAAIC60222.2024.10575567.

[66] S. Savaram, K. Goutham and K. Venkatasubramanian, ”Comparative Analysis of YOLO Models for Real-Time Safety Helmet Detection to Enhance Construction Site Safety Using Raspberry Pi,” 2025 International Conference on Next Generation Communication Informa- tion Processing (INCIP), Bangalore, India, 2025, pp. 513-518, doi: 10.1109/INCIP64058.2025.11019464.

[67] L. Duvvuri and R. G. J, ”SSD Framework in Raspberry Pi for Real-Time Object Detection in Autonomous Vehicles,” 2024 IEEE International Conference on Electronics, Computing and Communica- tion Technologies (CONECCT),

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26mar114

Bangalore, India, 2024, pp. 1-6, doi: 10.1109/CONECCT62155.2024.10677229.

[68] S. Duman, A. Elewi and A. Souag, ”Real-Time Mushroom Detection and Maturity Classification Using YOLO-Tiny on Raspberry Pi Plat- form,” 2025 9th International Symposium on Innovative Approaches in Smart Technologies (ISAS), Gaziantep, Turkiye, 2025, pp. 1-5, doi: 10.1109/ISAS66241.2025.11101938.

[69] El-Alami, Y. Nadir, and K. Mansouri, “A Hybrid Vehicle Tracking System for Low-power Embedded Devices,” in Proc. IEEE International Conference (published Jun. 28, 2024).

[70] S. N. Adu Tagoe, M. Elmir, and N. Amanquah, “Monitoring of Animal Movement using Computer Vision,” in 2024 IEEE 9th International Conference on Adaptive Science and Technology (ICAST), Accra, Ghana, 2024, pp. 1–6, doi: 10.1109/ICAST61769.2024.10856474.

[71] S. S. Yadav, S. Anand, A. M. D, D. S. Nikitha, and C. S. Thakur, “tinyRadar: LSTM-based Real-time Multi-target Human Activity Recog- nition for Edge Computing,” in 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, 2024, pp. 1–5, doi: 10.1109/ISCAS58744.2024.10558474.

[72] A. Messaoui, R. Louali, F. Demim, I. Zohir, et al.,“Detection and Tracking of Mobile Objects Using Hybrid Algorithms,” in Proceedings of the 6th International Conference on Electrical Engineering and Control Applications (ICEECA), Vol. 1, Nov. 2025, doi: 10.1007/978- 981-95-1109-9-27.

[73] Sridharan and G. M, ”A Lightweight Convolutional Network for Deep Learning-based Ball Manipulation by a Soccer Playing Robot,” 2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2024, pp. 1305- 1310, doi: 10.1109/ICACRS62842.2024.10841538.

[74] Z. Tang and J. Wang, “Research on real-time face recognition and tracking model based on embedded platform,” in Proc. SPIE/COS Photonics Asia, Nov. 21, 2025.